$$Q_\varepsilon(\mu, \mathbb{X}) \leqslant \sum_{a \in A} \frac{\binom{L-q-r}{\ell-q}}{\binom{L}{\ell}} H^{\ell-q,\, m-r}_{L-q-r}\left(\tfrac{\ell}{L}(m - \varepsilon k)\right), \tag{1}$$

1 Sources-based bound

Bound (1) is exact for some model sets of classifiers, but on experiments with real data it can be overestimated by several orders. In this section we derive a sharper bound that takes into account similarities between classifiers with incomparable error vectors.

Let A_{ij} be a set of all objects where a_j produces an error but a_i doesn't:

$$A_{ij} = \{x \in \mathbb{X} \,|\, I(a_i, x) = 0, I(a_j, x) = 1\}.$$

Lemma 1 *Suppose learning algorithm μ is a PessERM and all algorithms in A are arranged in ascending order with respect to the number of errors. Then for any sample $X \subseteq \mathbb{X}$*

$$[\mu X = a_i] = \left(\prod_{j=1}^{i-1}\Big[|X \cap A_{ji}| \leqslant |X \cap A_{ij}|\Big]\right)\left(\prod_{j=i+1}^{D}\Big[|X \cap A_{ji}| < |X \cap A_{ij}|\Big]\right). \tag{2}$$

This lemma gives a necessary and sufficient condition for choosing classifier a by learning algorithm μ. This condition cannot be used directly for computation of P_a because of its complexity, but it can be turned into necessary condition by exclusion of some multipliers. For example, if we keep only multipliers corresponding to upper neighbourhood of a and to all sources of SC-graph with error vectors comparable to a, then we can derive combinatorial bound (1). However, this bound takes in account only similarity between algorithms with comparable error vectors, which is quite restrictive. To improve this we propose to compare a with one sources that results in minimal contribution of a into bound.

Theorem 2 *Suppose learning algorithm μ is an ERM, S is a set of all sources of SC-graph and $A = \{a_1, \ldots, a_D\}$ is a set of classifiers. Then*

$$Q_\varepsilon(\mu, \mathbb{X}) \leqslant \sum_{i=1}^{D} \min_{s \in S}\left\{\sum_{t=0}^{T_{is}} \frac{C_q^t C_{L-u-q}^{\ell-u-t}}{C_L^\ell} \mathcal{H}^{\ell-u-t,\, m-q}_{L-u-q}\Big(\tfrac{\ell}{L}(m - \varepsilon k) - t\Big)\right\}, \tag{3}$$

where $u \equiv u(a_i)$, $q \equiv q(a_i)$, $m \equiv m(a_i)$, $T_{is} = \min(|A_{is}|, |A_{si}|)$.

2 Overfitting bounds estimation with random walks

All combinatorial bounds have form

$$Q_\varepsilon \leqslant B_\varepsilon = \sum_{a \in A} b(a),$$

which involves summation over all classifiers. In practice this is intractable since the number of classifiers can be enormously large. It is known that only classifiers from lower layers of SC-graph make significant contribution to combinatorial bounds, but even lower layers can be too large. To overcome this difficulty we propose to sample classifiers from SC-graph and use them to estimate bound B_ε. This idea is formalized by random walk techniques.

Denote SC-graph by $G = (V, E)$. The basic purpose of random walk is to generate a sequence of vertices from the graph such that probabilities of picking every vertex are known. One way to achieve this is to use simple random walk (SRW) which starts from some vertex and then walks on the graph, choosing next vertex uniformly from the neighbourhood of current vertex. It is known that for connected and non-bipartite graphs the frequency of vertex v appearance in SRW walk converges to the following *stationary* distribution:

$$\pi(v) = \frac{\deg(v)}{2|E|}.$$

However, simple random walk explores the graph very slowly and it may take too many iterations to generate a representative sample of classifiers. This can be improved by generating several initial vertices and starting a simple random walk from each of them. This idea is formalized in *Frontier Sampling (FS)* algorithm (Ribeiro and Towsley, 2010), which performs several parallel dependent

random walks, and generates a sequence of vertices with the same stationary distribution π as in simple random walk.

Let $a_1, \ldots, a_n$ be a sample of classifiers from A generated by SRW or FS algorithm. It can be derived from the Strong Law of Large Numbers for Markov chains (Roberts and Rosenthal, 2004) that the following gives an unbiased estimator for B_ε:

$$\hat{B}_\varepsilon = |A| \frac{\sum_{i=1}^{n} b(a_i)/\deg(a_i)}{\sum_{i=1}^{n} 1/\deg(a_i)}. \tag{4}$$

The proof is given in Appendix.

3 Compositions of linear classifiers

Set of linear classifiers. Consider a binary classification problem with labels $y_i \in \{-1, +1\}$, $i = 1, \ldots, L$ assigned to each object $x_i \in \mathbb{X} \subset \mathbb{R}^d$ respectively. Consider a set of unbiased linear classifiers

$$a(x; w) = \operatorname{sign}\langle w, x\rangle,$$

where $w \in \mathbb{R}^d$ is a real vector of weights. Each linear classifier corresponds to a hyperplane such that objects are labeled as "+1" on one side of the hyperplane and as "−1" on the other.

Traversal of SC-graph. We will use random walks to estimate bounds of overfitting, so an algorithm for finding neighbourhood of a linear classifier is needed. Our proposed algorithm is based on geometrical arrangements framework and described in details in appendix.

Experimental design. We use combinatorial bounds to build a composition of linear classifiers by simple majority voting:

$$a(x) = \operatorname{sign} \sum_{i=1}^{p} \operatorname{th}\langle w_i, x\rangle. \tag{5}$$

Each of basic classifiers is trained on subsample choosen by ComBoost algorithm, which discards objects with too large and too small margins.

After choosing subsample we do feature selection by greedy incremental search. We choose the feature set with minimal inverted bound on every step; the inverted bound is computed as follows:

1. A linear classifier is trained by SVM.
2. We step down from the vertex corresponding to this classifier to any source of SC-graph.
3. We explore all SC-graph layers up to $(m_0 + 3)$-th for other sources, where m_0 is the layer of the vertex found on previous step.
4. We sample 2000 vertices by Frontier Sampling algorithm starting from all found sources.
5. The bound (3) or (1) is estimated using (4). Then estimated bound is inverted to get our feature set quality criterion:
$$Q_c = \nu(a_0, X) + \varepsilon\,(1/2)\,,$$
where a_0 is the classifier with minimal number of errors.

We compare described method with the following modifications:

- Instead of inverted combinatorial bound we can use empirical risk $\nu(a_0, X)$ as a criterion for feature selection. Comparison with this alternative can show that using overfitting bounds gives advantage over methods that do not consider overfitting explicitly.
- Instead of inverted combinatorial bound we can use a cross-validation bound as a criterion for feature selection. Comparison with this alternative can show that combinatorial bounds are better then empirical Monte-Carlo bounds.
- Instead of building a composition using ComBoost and feature selection we can build a two-layer neural network with hyperbolic tangent activation function using backpropagation. This alternative is interesting because such neural network gives a classifier of the form (5).

Experimental results. We tested our approach on 4 real data sets from UCI. For each data set we randomly chose 200 object for training sample. The results are shown in table 1.

	Wine Quality	Statlog	Waveform	Faults
ComBoost and feature selection based on empirical risk $\nu(\mu, \mathbb{X})$	$64,70$	$84,92$	$84,78$	$73,39$
Two-layer neural network	$72,06$	$85,41$	$86,79$	$74,51$
ComBoost and feature selection based on cross-validation bound	$71,06$	$85,26$	$86,38$	$75,76$
ComBoost and feature selection based on bound (1)	$69,48$	$86,26$	$85,77$	$\mathbf{77,81}$
ComBoost and feature selection based on bound (3)	$\mathbf{74,68}$	$\mathbf{86,75}$	$\mathbf{86,91}$	$74,03$

Table 1: Experimental results for 4 data sets from UCI repository. For each algorithm and each data set a test quality in percents is given.

4 Appendix

4.1 Proof of sources-based bound.

Proof of Lemma 1. We suppose that is there are few classifiers with the minimal number of errors on training set X and the maximal number of errors on test set $\bar{X}$, then PessERM μ chooses the classifier with the largest index in A. Let $A = \{a_1, \ldots, a_D\}$.

Note that if $|X \cap A_{ji}| > |X \cap X_{ij}|$ holds, then a_j makes more error on training set X than a_i. Therefore a_i cannot be chosen by learning algorithmμ.

Suppose now that $|X \cap A_{ji}| = |X \cap X_{ij}|$ and $j > i$. It follows from $|X \cap A_{ji}| = |X \cap X_{ij}|$ that classifiers a_i and a_j make the same number of errors on training set X: $n(a_i, X) = n(a_j, X)$. From $j > i$ follows $m(a_j) \geqslant m(a_i)$ since classifiers are arranged in ascending order with respect to the number of errors. Then

$$n(a_j, \bar{X}) = m(a_j) - n(a_j, X) \geqslant m(a_i) - n(a_i, X) = n(a_i, \bar{X}).$$

In other words, classifiers a_i and a_j make the same number of errors on training set and a_j make not less errors on test set and has larger index. That means that a_i cannot be chosen by learning algorithm μ.

So we found necessary conditions for a_i to be chosen by learning algorithm μ:

- $|X \cap A_{ji}| \leqslant |X \cap X_{ij}|$ if $j < i$;
- $|X \cap A_{ji}| < |X \cap X_{ij}|$ if $j > i$;

for all $j = 1, \ldots, D$ where D is the number of classifiers in A. Then the following holds:

$$[\mu X = a_i] \leqslant \left(\prod_{j=1}^{i-1} \Big[|X \cap A_{ji}| \leqslant |X \cap A_{ij}| \Big] \right) \left(\prod_{j=i+1}^{D} \Big[|X \cap A_{ji}| < |X \cap A_{ij}| \Big] \right).$$

Now we prove that this conditions are also sufficient. Suppose that this conditions hold. Consider an arbitrary classifier $a_j \neq a_i$. If $|X \cap A_{ji}| < |X \cap X_{ij}|$ then a_j make more errors on training set than a_i and cannot be chosen by learning algorithm μ. If $|X \cap A_{ji}| < |X \cap X_{ij}|$ then $j < i$ and $m(a_j) \leqslant m(a_i)$. So $n(a_j,' barX) \leqslant n(a_i, \bar{X})$ and a_j cannot be chosen by μ. Then any classifier $a_j \neq a_i$ cannot be chosen by μ. Then $[\mu X = a_i]$ and

$$[\mu X = a_i] \geqslant \left(\prod_{j=1}^{i-1} \Big[|X \cap A_{ji}| \leqslant |X \cap A_{ij}| \Big] \right) \left(\prod_{j=i+1}^{D} \Big[|X \cap A_{ji}| < |X \cap A_{ij}| \Big] \right).$$

■

Algorithm 4.1 Simple Random Walk.

Require: Graph $G = (V, E)$; iteration number N; initial vertex v_0;
Ensure: Sample of classifiers $v_1, v_2, \ldots, v_N$;

1: **for** $i = 1, \ldots, N$ **do**
2: **with probability** $\frac{1}{2}$
3: pick a vertex v' from uniform distribution on $\{v' \in V \mid (v_{i-1}, v') \in E\}$;
4: $v_i := v'$;
5: **otherwise**
6: $v' := v$; $v_i := v$;

Proof of Theorem 2. At first we derive an upper bound on $[\mu X = a_i]$ using Lemma 1 by taking into account similarities between a_i and its upper neighbourhood $C^+(a_i)$ and some source a_s:

$$\begin{aligned}
[\mu X = a_i] \leqslant \Big([s \leqslant i]\Big[|A_{si} \cap X| \leqslant |A_{is} \cap X|\Big] + [s > i]\Big[|A_{si} \cap X| < |A_{is} \cap X|\Big]\Big) \times \\
\times \prod_{j:\ a_j \in C^+(a_i)} \Big[|A_{ji} \cap X| < |A_{ij} \cap X|\Big] \leqslant \\
\leqslant \Big[|A_{si} \cap X| \leqslant |A_{is} \cap X|\Big] \prod_{j:\ a_j \in C^+(a_i)} \Big[|A_{ji} \cap X| < |A_{ij} \cap X|\Big] \leqslant \\
\leqslant \Big[|A_{si} \cap X| \leqslant |A_{is}|\Big] \prod_{j:\ a_j \in C^+(a_i)} \Big[|A_{ji} \cap X| < |A_{ij} \cap X|\Big] = \\
= \Big[|A_{si} \cap X| \leqslant |A_{is}|\Big] \prod_{j:\ a_j \in C^+(a_i)} \Big[|A_{ij} \cap X| > 0\Big]
\end{aligned}$$

So it is necessary for classifier a_i to be chosen by learning algorithms μ that all objects from $\bigcup_{a_j \in C^+(a_i)} A_{ij}$ are in training set and that there are no more than $|A_{is}|$ objects from A_{si} in X. Then it is easy to derive the following bound:

$$\begin{aligned}
&\mathsf{P}[\mu X = a_i]\mathsf{P}\big[\delta(a_i, X) \geqslant \varepsilon \;\big|\; \mu X = a_i\big] \leqslant \\
&\qquad \leqslant \sum_{t=0}^{\min(|A_{is}|, |A_{si}|)} \frac{C^t_{|A_{si}|} C^{\ell-u-t}_{L-u-|A_{si}|}}{C^\ell_L} \mathcal{H}^{\ell-u-t,\ m-|A_{si}|}_{L-u-|A_{si}|} \left(\frac{\ell}{L}(m - \varepsilon k) - t\right).
\end{aligned}$$

To get the final bound we use the law of total probability and for each classifier a_i we choose the source a_s that results in minimal contribution of a_i into bound:

$$\begin{aligned}
&Q_\varepsilon(\mu, \mathbb{X}) = \sum_{i=1}^{D} \mathsf{P}[\mu X = a_i]\mathsf{P}\big[\delta(a_i, X) \geqslant \varepsilon \;\big|\; \mu X = a_i\big] \leqslant \\
&\qquad \leqslant \sum_{i=1}^{D} \min_{s \in S} \left\{ \sum_{t=0}^{T_{is}} \frac{C^t_{|A_{si}|} C^{\ell-u-t}_{L-u-|A_{si}|}}{C^\ell_L} \mathcal{H}^{\ell-u-t,\ m-|A_{si}|}_{L-u-|A_{si}|} \left(\frac{\ell}{L}(m - \varepsilon k) - t\right) \right\}.
\end{aligned}$$

■

4.2 Random walk algorithms.

We give here random walk algorithms for convenience, see alg. 4.1 and 4.2. Note that both algorithms produce a sequence of classifiers with stationary distribution only given that graph G is connected and non-bipartite. However, SC-graph is always multipartite and therefore it is bipartite. A very simple way to solve this problem is to use *lazy* random walks (Lovasz, 1993) that doesn't make any move on each step with some probability p (usually $p = 1/2$). This modification is made both in algorithms 4.1 and 4.2.

4.3 Bound estimation.

Suppose we have a sample of classifiers $a_1, \ldots, a_n$ generated by algorithms 4.1 or 4.2, so it is known that this sequence has a stationary distribution $\pi(a) = \deg(a)/2|E|$. Our purpose is to estimate the bound $B_\varepsilon = \sum_{a \in A} b(a)$ using this sample.

Algorithm 4.2 Frontier Sampling.

Require: Graph $G = (V, E)$; iteration number N; initial vertices $P = (v^1, \ldots, v^s)$;
Ensure: Sample of classifiers $v_1, v_2, \ldots, v_N$;

for $i = 1, \ldots, N$ **do**
  choose $v \in P$ with probability $\frac{\deg(v)}{\sum_{u \in P} \deg(u)}$;
  with probability $\frac{1}{2}$
    pick a vertex v' from uniform distribution on $\{v' \in V \mid (v, v') \in E\}$;
    $v_i := v'$;
  otherwise
    $v' := v$; $v_i := v$;
  Replace in P vertex v with v';

The Strong Law of Large Numbers for Markov chains (Roberts and Rosenthal, 2004) tells that if random walk has a stationary distribution π then

$$\mu_n(f) = \frac{1}{n} \sum_{i=1}^{n} b(a_i) \xrightarrow[n \to \infty]{\text{a. s.}} \sum_{a \in A} b(a)\pi(a).$$

To get an estimate for B_ε we should use the weight function $w(v) = \frac{1}{\pi(a)}$; in this case

$$\mu_n(wf) = \frac{1}{n} \sum_{i=1}^{n} w(a_i)b(a_i) = \frac{1}{n} \sum_{i=1}^{n} \frac{b(a_i)}{\pi(a_i)} \xrightarrow[n \to \infty]{\text{a. s.}} \sum_{a \in A} \frac{b(a)}{\pi(a)}\pi(a) = \sum_{a \in A} b(a) = B_\varepsilon,$$

so $\mu_n(wf)$ gives a consistent estimator for B_ε. However, expression for $\mu_n(wf)$ contains number of edges $|E|$ which is usually unknown. The solution is to use a slightly different estimator:

$$\hat{B}_\varepsilon = |A| \frac{\mu_n(wf)}{\mu_n(w)} = |A| \frac{\frac{1}{n} \sum_{i=1}^{n} \frac{2|E|}{\deg(a_i)} b(a_i)}{\frac{1}{n} \sum_{i=1}^{n} \frac{2|E|}{\deg(a_i)}} = |A| \frac{\sum_{i=1}^{n} b(a_i)/\deg(a_i)}{\sum_{i=1}^{n} 1/\deg(a_i)}.$$

To prove that this bound is consistent note that

$$\mu_n(wf) \xrightarrow[n \to \infty]{\text{a. s.}} \sum_{a \in A} b(a) = B_\varepsilon$$
$$\mu_n(w) \xrightarrow[n \to \infty]{\text{a. s.}} \sum_{a \in A} 1 = |A|$$

Note that $|A|$ is known, for example, for the set of linear classifiers.

4.4 Equivalence classes of linear classifiers.

We will use geometrical arrangements framework (Agarwal and Sharir, 1998) to analyze equivalence classes of the set of linear classifiers.

Denote the parameter space of our classifier set as $\mathbb{W} \equiv \mathbb{R}^d$. Let $\mathbb{X}$ be in general position.

Our goal is to describe classes of hyperplanes that have the same error vectors and to find an effective algorithm for determining the neighbourhood of linear classifier in SC-graph. Instead of working with points from $\mathbb{X}$ and hyperplanes that separate them, it's more convenient to switch to dual space, where each linear classifier corresponds to a single point, and each object from $\mathbb{X}$ corresponds to a hyperplane. To do this, associate each object $x_i \in \mathbb{X}$ to a hyperplane in a parameter space $\mathbb{W}$: $h_i = \{w \in \mathbb{W} : \langle w, x_i \rangle = 0\}$; also associate every hyperplane $\{x \in \mathbb{R}^d : \langle w, x \rangle = 0\}$ to a point w in a parameter space. Each h_i dissects $\mathbb{W}$ into two halfspaces: positive $h_i^+ = \{w \in \mathbb{W} : I(a(\cdot; w), x_i) = 0\}$ and negative $h_i^+ = \{w \in \mathbb{W} : I(a(\cdot; w), x_i) = 1\}$, corresponding to classifiers giving a correct/incorrect answer on x_i respectively.

The set of hyperplanes $\mathbb{H} = \{h_i : i = 1, \ldots, L\}$ partitions $\mathbb{W}$ into convex polyhedrons called *cells*. Formally each cell is a closure of some connected component of the set $\mathbb{W} \setminus \mathbb{H}$. Each cell is convex because it can be represented as an intersection of some halfspaces. The described partition of $\mathbb{W}$ into cells is called *cell arrangement*.

The main property of cell arrangement is that each classifier a bijectively corresponds to some cell C_a, so all linear classifiers with same error vectors lie in one cell. If two classifiers a_1 and a_2

are connected by an edge in SC-graph, then their error vectors differ only in one position. This means that cells C_{a_1} and C_{a_2} share the $(d-1)$-dimensional face. Consequently, cells corresponding to neighbour algorithms have a common vertex.

Our algorithm for finding neighbourhood of a classifier at first determines all cells that have a common vertex with cell corresponding to given classifier and then checks for each of them whether their error vector differ from given classifier's vector only in one position.

At first we describe how to find a vertex of a cell given a point inside this cell. Let $z^0 = (z_1^0, \ldots, z_d^0)$ be any point inside cell C_{a_0}. We will find a vertex of C_{a_0} in d steps. At the first step we try to project z^0 on each hyperplane from $\mathbb{H}$ and find such projection that its error vector is the same as of z^0. Denote the hyperplane that satisfies that requirement as h^1 and the projected point as z^1. At the second step we try to project z^1 on each flat from $\mathbb{H} \cap h^1 = \{h_1 \cap h^1, \ldots, h_L \cap h^1\} \setminus h^1$, and find projection with the same error vector. After d steps we will find a point z^d that lies in intersection of d hyperplanes from $\mathbb{H}$, which means that z^d is a vertex of cell arrangement, and that has the same error vector as z^0, which means that z^d still lies in cell C_{a_0}.

The second stage of our algorithm is to find *all* vertices of the cell C_{a_0}. We already have one vertex v that lies on intersection of d hyperplanes; denote that set of hyperplanes as H_v. It can be shown that if vertices v and u of arrangement are connected with an edge, then $|H_v \cap H_u| = d-1$, so to find all vertices connected by edge with v it's sufficient to try all ways to change one hyperplane in H_v. Then we do breadth-first search to find all vertices of the cell.

Let $G = (V, E)$ be a graph where vertices V correspond to vertices of cell and edges E correspond to edges of this cell. It is known (Balinski, 1961) that this graph is connected, so BFS will find all vertices.

When all vertices of C_{a_0} are found, it's easy to find all its neighbours. Let v be a vertex that lies on intersection of hyperplanes from H_v. Recall that each hyperplane $h_i \in H_v$ corresponds to some object $x_i \in \mathbb{X}$. Then C_{a_0} has $(d-1)$-dimensional faces corresponding to all such objects, which means that a_0 has neighbour algorithms with error vectors different on these objects.

References

P. K. Agarwal and M. Sharir. Arrangements and their applications. *Handbook of Computational Geometry*, 49–119, 1998.

M. L. Balinski. On the graph structure of convex polyhedra in n-space. *Pacific Journal of Mathematics*, 11:431–434, 1961.

L. Lovasz. Random Walks on Graphs: A Survey. *Combinatorics, Paul Erdos is Eighty*, 2(1):1–46, 1993.

B. Ribeiro and D. Towsley. Estimating ans sampling graphs with multidimensional random walks. *10th Conf. on Internet Measurement*, 390–403, 2010.

G. O. Roberts and J. S. Rosenthal. General state space Markov chains and MCMC algorithms. *Probability Surveys*, 1:20–71, 2004.